A 2nd GENERATION PARALLEL ADVANCING FRONT GRID GENERATOR

Rainald Löhner

CFD Center, Dept. of Computational and Data Science
M.S. 6A2, College of Science, George Mason University
Fairfax, VA 22030-4444, USA
rohner@gmu.edu

Summary. A scalable, parallel advancing grid generation technique has been developed for complex geometries and meshes with large size variations. The key innovation compared to previous techniques is the use of a domain-defining grid that has the same fine surface triangulation as the final mesh desired, but a much coarser interior mesh. In this way, the domain to be gridded is uniquely defined, overcoming a shortcoming of previous approaches. This domain-defining grid is then partitioned according to the estimated number of elements to be generated, allowing for a balanced distribution of work among the processors. The domain defining grid is also used to redistribute the elements and points after grid generation, and during the subsequent mesh improvement.
Timings show that the proposed approach is scalable and able to produce large grids of high quality in a modest amount of clocktime.
With the proposed parallel grid generator, a major impediment to a completely scalable simulation pipeline (grid generation, solvers, post-processing) has been removed, opening the way for truly large-scale computations using unstructured, body-fitted grids.

1 INTRODUCTION

The widespread availability of parallel machines with hundreds of thousands of cores and very large memory, solvers that can harness the power of these machines, and the desire to model in ever increasing detail geometrical and physical features has led to a steady increase in the number of points and elements used in field solvers. During the 1990s, grids in excess of 10^7 elements became common for production runs in Computational Fluid Dynamics (CFD) [4, 5, 28, 69, 51] and Computational Electromagnetics (CEM) [17, 54]. This tendency has continued during the first decade of the 21st century, roughly following Moore's law, i.e. gridsizes have increased by an order of magnitude every 5 years. Presently, grids in of the order of 10^9 elements are commonly used for leading edge applications in the aerospace, defense, automotive, naval,

energy and electromagnetics sectors.
While many solvers have been ported to distributed parallel machines, grid generators have, in general, lagged behind. One can cite several reasons for this:

a) For many applications the CPU requirements of grid generation are orders of magnitude less than those of field solvers, i.e. it does not matter if the user has to wait several hours for a grid;

b) (Scalar) grid generators have achieved a high degree of maturity, generality and widespread use, leading to the usual inertia of workflow ('modus operandi') and aversion to change;

c) In recent years, low-cost machines with few cores but very large memories have enabled the generation of large grids with existing (scalar) software; and

d) In many cases it is possible to generate a mesh that is twice (2^d times) as coarse as the one desired for the simulation. This coarse mesh is then h-refined globally. Global h-refinement is easily ported to multicore and/or distributed memory machines. Moreover, many field solvers offer h-refinement as an option. With only 1 level of h-refinement a mesh of 125 Mels increases to 1 Bels, and to 15.6 Mels with two levels of h-refinement.

For applications where remeshing is an integral part of simulations, e.g. problems with moving bodies [37, 52, 53, 6, 30, 43, 24] or changing topologies [7, 8], the time required for mesh regeneration can easily consume a significant percentage of the total time required to solve the problem. This percentage increases drastically if the grid generation portion is not completely parallelized. Faced with this situation, a number of efforts have been reported on parallel grid generation [38, 14, 62, 15, 55, 20, 10, 56, 69, 9, 11, 22, 61, 47, 12, 13, 29, 26, 64, 1, 2].
The two most common ways of generating unstructured grids are the Advancing Front Technique (AFT) [57, 58, 35, 36, 59, 60, 27, 19, 42, 47] and the Generalized Delaunay Triangulation (GDT) [3, 21, 67, 68, 50, 10, 9, 12, 64, 1]. The AFT introduces one element at a time, while the GDT introduces a new point at a time. Thus, both of these techniques are, in principle, scalar by nature, with a large variation in the number of operations required to introduce a new element or point. While coding and data structures may influence the scalar speed of the 'core' AFT or GDT, one often finds that for large-scale applications, the evaluation of the desired element size and shape in space, given by background grids, sources or other means [48] consumes the largest fraction of the total grid generation time. Furthermore, the time required for mesh improvements (and any unstructured grid generator needs them) is

in many cases higher than the core AFT or GDT modules. Typical speeds for the complete generation of a mesh (surface, mesh, improvement) on current Intel Xeon chips with 3.2GHz and sufficient memory are of the order of 0.5-2.0 Mels/min. Therefore, it would take approximately 2,000 minutes (i.e. 1.5 days) to generate a mesh of 10^9 elements. Assuming perfect parallelization, this task could be performed in the order of a minute on 2,000 processors, clearly showing the need for parallel mesh generation.

Unstructured grid generators based on the AFT may be parallelized by invoking distance arguments, i.e., the introduction of a new element only affects (and is affected by) the immediate vicinity. This allows for the introduction of elements in parallel, provided that sufficient distance lies between them.

Nearly two decades ago (when useful distributed memory parallel machines first appeared) Löhner, Camberos and Merriam [38] introduced a parallel AFT for 2-D applications. This was extended shortly afterwards to 3-D by Shostko [62]. The spatial distribution of work was based on the subdivision of a relatively fine background grid. While used for some demonstration runs, this scheme was not general enough for a production environment. The background grid had to be adapted in order to be sufficiently fine for a balanced workload. As only background grid elements covering the domain to be gridded were allowed, complex in/out tests had to be carried out to remove refined elements lying outside the domain to be gridded. Furthermore, element size specified at CAD entities could not be 'propagated' into the domain, as is the case in the scalar AFT, disabling an option favoured by many users and rendering many grid generation data sets unusable. The otherwise positive experience gained with this parallel AFT, and the rise of shared-memory machines, prompted the search for a more general parallel AFT. The key requirement was a parallel AFT that modified the mature, scalar AFT as little as possible, while achieving significant speedups on common parallel machines. This led to a shared-memory parallel AFT (based on OpenMP) that applied the parallelism at the level of the current front, and not globally [47]. This scheme has been used for more than a decade, and has yielded a means of speeding up grid generation by an order of magnitude. Given that the parallelism is invoked at the level of the front, the achievable scalability is clearly limited.

The advent of machines with hundreds of thousands of processors has led to a re-evaluation of the parallel grid generation options. It is clear that for machines with such a high number of processors, every effort has to be made to extract the maximum parallelism possible at every stage of the grid generation. This means that the **parallelism should not be front-based, but volume-based**. The easiest form of achieving volume-based parallelism is by using a grid to define the regions to be meshed by each processor. Optimally, this domain-defining grid (DDG) should have the same surface triangulation as the desired, fine mesh, but could be significantly coarser in the interior. In this way, the definition of the domain to be gridded is unique, something that is notoriouly difficult to achieve by other means (such as background grids, bins or octrees). This domain-defining grid is then split so that in each

subdomain a similar number of elements is generated.

2 DESIRED FEATURES FOR THE NEXT-GENERATION PARALLEL MESHER

Before describing the proposed 2nd generation parallel grid generator, we list the main characteristics such a tool should offer:

- Use of the (fine) surface mesh specified by the user: this means that no global h-refinement can/needs to be used; the key assumption is that this surface mesh can not be coarsened and then subsequently h-refined;
- Maximum re-use of existing scalar grid generation software: it takes a decade to build a robust, production-quality 3-D grid generator; therefore, being able to reuse existing software would be extremely desirable;
- AFT or GDT: the two main ways of generating general unstructured grids are the advancing front technique (AFT) and the Generalized Delaunay Triangulation (GDT); the parallel grid generator should be able to use any of these techniques;
- Maximum re-use of existing grid generation features/options, such as:
  - mesh size specified via background grid;
  - mesh size specified via sources;
  - mesh size specified via CAD entities (points, lines, surfaces, domains);
  - optimal space-filling tet options;
  - link to boundary layer grids;
- Use of multicore parallel machines: given that massively parallel machines will be composed of multicore chips, it would be highly desirable to exploit effectively this type of architecture.

3 2nd-GENERATION PARALLEL MESHER

The key idea of the proposed 2nd-generation parallel mesh generator is the use of two levels of grid generation:

- One to subdivide space into regions that will generate approximately the same number of elements, and
- One that performs the parallel grid generation.

These two tasks, could, in principle, be carried out with different grid generation techniques/codes, making the approach very general. The procedure is shown conceptually in Figures 1a-c.

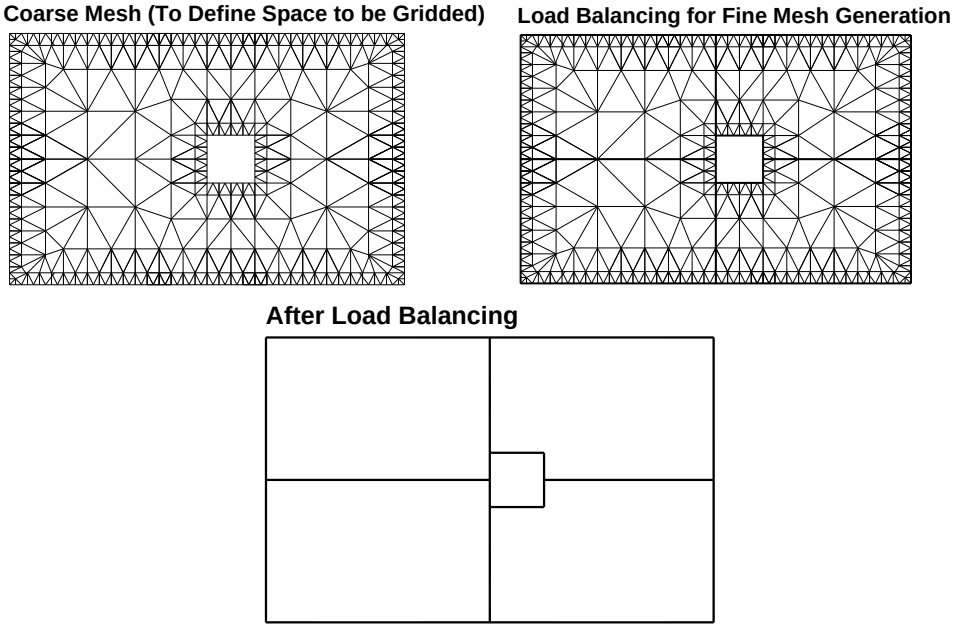


Figure 1 Splitting of Domain Defining Grid

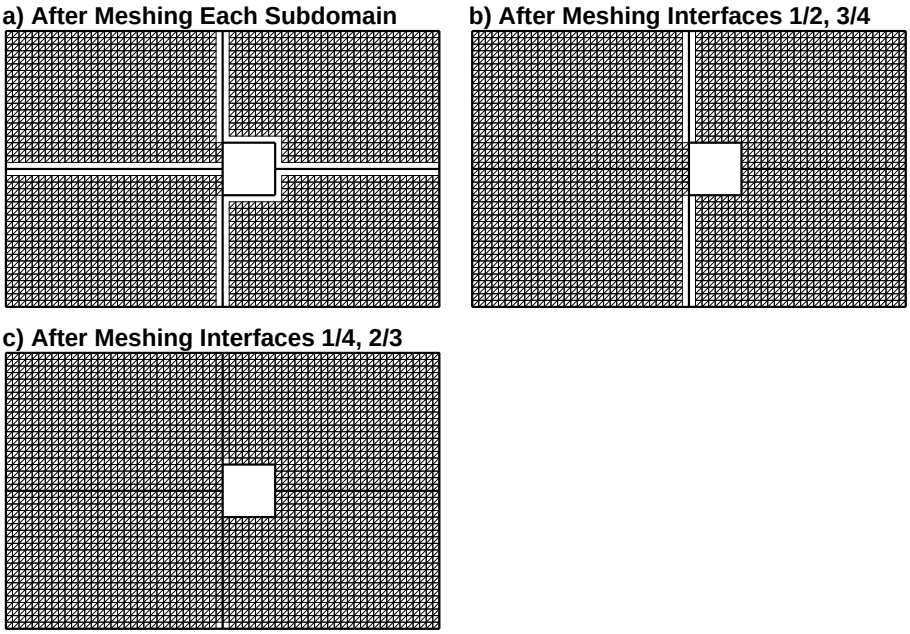


Figure 2 Parallel Grid Generation Technique

4 BASIC ADVANCING FRONT TECHNIQUE

Before going on, we recall for the sake of clarity and completeness the main algorithmic steps of the advancing front technique:
Assume given:

- AG1: A definition of the spatial variation of element size, stretchings, and stretching directions for the elements to be created. In most cases, this is accomplished via a combination of background grids, sources and CAD-based information [48].
- AG2: A watertight, topologically consistent triangulation that is conmesurate with the desired element size and shape. This is the so-called initial front.
- AG3: The generation parameters (element size, element stretchings and stretching directions) for the faces of the initial front.

Then:
`While` there are active faces left in the front:

- AF1: Select the next face `ifout` to be deleted from the front; in order to avoid large elements crossing over regions of small elements, the face forming the smallest new element is selected;
- AF2: For the face to be deleted:
  - AF2.1: Select a 'best point' position for the introduction of a new point `ipnew`;
  - AF2.2: Determine whether a point exists in the already generated grid that should be used in lieu of `ipnew`; if there is such a point, set this point to `ipnew`;
  - AF2.3: Determine whether the element formed with the selected point `ipnew` crosses any given faces; if it does, select a new point as `ipnew` and try again; if none can be found: skip `ifout`;
- AF3: Add the new element, (point, faces) to their respective lists;
- AF4: Find the generation parameters for the new faces;
- AF5: Delete the known faces(s) from the list of faces;

`End While`
Individual aspects of the technique (such as optimal data structures for speed, robust checking of face intersections, filtering techniques to avoid unnecessary work, etc.) may be found in [40, 48].

5 GENERATION OF THE DOMAIN DEFINING GRID (STEP 1)

Given that the number of elements and points decreases with the 3rd power of the element size, a mesh with elements whose side-lengths are n times as large as the desired one will only contain n^{-3} elements as the (fine) mesh desired.

The idea is then to generate, starting from the fine surface mesh, a mesh whose elements are considerably larger than the grid desired. A factor of $n = 10$ will lead to a mesh that is generated in roughly 1/1000-th of the time required for the fine mesh. For $n = 20$, the factor is 1/8000. The mesh obtained, though, conforms to the general size distribution required by the user, i.e. is completely general. Moreover, it allows to determine exactly and easily which regions of space need to be gridded (one of the problematic aspects of earlier parallel grid generators [38, 62, 47]). In the following, we will denote this mesh as the domain-defining grid DDG.
In order to generate the DDG, the changes required to the basic advancing front technique are restricted to the desired element size, which has to increase rapidly as elements are generated in the volume:

- The generation parameters for the initial front (Step AG3 above) are multiplied by the increase factor c_i allowed for each face removed from the front. Typical values are: $c_i = 1.5 - 1.7$.
- When a new point is added to the front, the grid generation parameters of the points belonging to the face being removed `ifout` are multiplied by c_i and used instead of the usual ones (which are obtained from the background grid and sources, see step AF4 above).

Note that as the advancing front technique always removes the face generating the smallest element from the front, no incompatibilities in element size appear when these changes are invoked. Thus, the generation of the DDG is of the same robustness as the basic underlying scalar AFT.
In practice one observes that the total number of extra points required to fill up the complete volume is of the order of the points on the boundary while element quality does not suffer.

6 LOAD BALANCING OF THE DOMAIN DEFINING GRID (STEP 2)

Given the DDG, the next task is to subdivide this mesh so as to obtain regions in which roughly the same numbers of elements will be generated. A number of load balancing techniques and codes have been developed over the last two decades [66, 23, 65, 39, 31, 32]. In principle, any of these can be used in order to obtain the subdivision required. For the results shown here, we used (FESPLIT), which offers the possibility of subdividing grids based on the advancing front/greedy, recursive coordinate/moment bisection, or via spacefilling curves. Once an initial subdivision is obtained, FESPLIT improves the load balance (e.g. surface to volume ratios, continuity of subdivisions, etc.) using a diffusion technique.

7 GENERATION OF THE FINAL MESH (STEP 3)

Once the subdivision of space is obtained, the mesh is generated in parallel. The technique used here is the 'inside-out' procedure first described in [38, 62], and consists in 4 passes, which are exemplified in Figure 2.

- Pass 1: The zones inside the subdivision domains are generated in parallel (see Figure 2a);
- Pass 2: The zones bordering the regions, which are left empty after pass 1, are meshed, in parallel, by pairing two domains at a time; by using a colouring technique, most of these inter-domain regions can be meshed completely in parallel (see Figure 2b);
- Pass 3: The zones bordering more than two regions (groups of domains), which are left empty after pass 2, are meshed, in parallel, by combining three or more domains at a time; as before, most of these inter-domain regions can be meshed completely in parallel by using a colouring technique;
- Pass 4: If required, the remaining regions are meshed on processor 1.

The following changes to the basic advancing front technique are required in order to obtain a reliable parallel meshing algorithm:

- If any of the points of the face to be removed lies outside the local DDG, the face is marked as prohibited and skipped;
- If the 'best point' position for the introduction of a new point lies outside the local DDG, the face is marked as prohibited and skipped;
- If any of the edges of the face to be removed `ifout` lies outside the local DDG, the face is marked as prohibited and skipped; this test is carried out by using a neighbour to neighbour traversal test between the points of the edge;
- If `ipnew` is on the both the inter-processor boundary of the DDG and the actual surface of the domain: the face is marked as prohibited and skipped;
- When assigning the faces and points to the local DDG, a conservative approach is taken; i.e. should an active front point coincide with the points of the DDG, all the surrounding DDG elements are tested to see if the point should be assigned to the present domain.

It is important to emphasize that all data is kept local. The list of elements and point being generated, the active front, and all other arrays are stored in the processor where they are being generated.

8 REDISTRIBUTION OF THE MESH (STEP 4)

After the parallel advancing front has completed the mesh, the pieces generated in each of the individual passes will be scattered among the different processors. This is particularly the case if a 4th Pass was required. In order to

arrive at a consistent mesh, the elements and points need to be redistributed and doubly defined points need to be removed.
Given that for each point the host element in the DDG is known, and that for each element of the DDG the processor it has been assigned to is also known, it is an easy matter so send the elements and the associated points to the processors they need to be. Each element is sent (if required; the majority already reside in the memory of the target processor) to the lowest processor assigned via points from the DDG. Doubly defined points are removed using an octree, so that this operation has $O(N \log(N))$ complexity.

The next step is to find the correlation between the points of neighbouring processors. In order to keep the procedure as general as possible, the following algorithmic steps are taken:

- The bounding box of each domain is computed;
- The bounding boxes of other domains that overlap the bounding box of each domain are determined; this determins a list of possible neighbouring domains;
- Each pair of possible neighbouring domains is tested in depth using octrees; in this way, the lists of neighbouring domains and points are obtained (so-called send/receive lists).

9 MESH IMPROVEMENT (STEP 5)

After the generation of the mesh using the parallel advancing front technique (or any other technique for that matter) has been completed, the mesh quality is improved by a combination of several algorithms, such as:

- Diagonal swapping,
- Removal of Bad Elements,
- Laplacian/Elasticity smoothing, and
- Selective mesh movement.

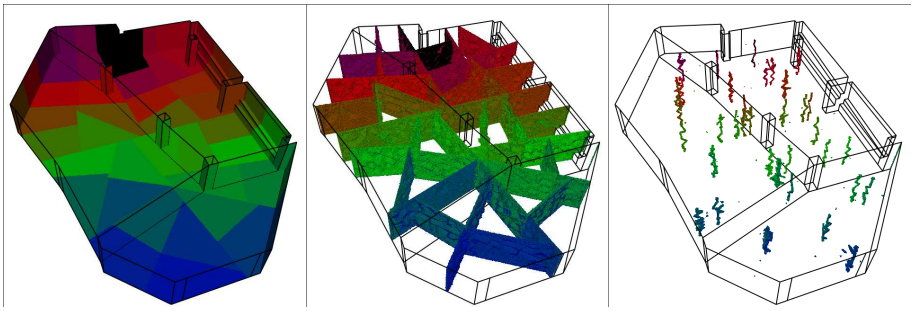
One should emphasize that mesh improvement may require CPU times that are comparable to those required by the basic grid generation technique, making it imperative to fully parallelize this necessary step as well. All of these procedures have been implemented and are running in parallel (shared locally via OMP and distributed globally via MPI). At the boundaries between processors, the procedures listed above would require a considerable amount of testing and information transfer. For this reason, it was decided not to allow any changes for the external faces of each subdomain. In order to improve the mesh in these regions as well, the DDG is redistributed among processors for a second time. The first distribution is taken as a starting point. Then, 1-2 extra layers of elements are added to the each domain `idomn` from the neighbouring domains `jdomn` for which `idomn < jdomn`. The elements of the generated (and smoothed) mesh are then redistributed as before.

10 EXAMPLES

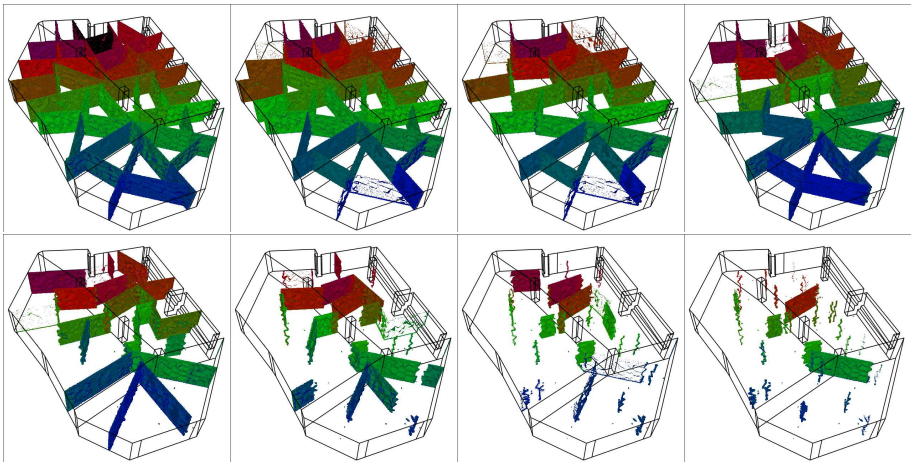
The 2nd generation parallel grid generator described above has been in operation for approximately a year. It is still undegoing considerable changes and improvements, so the numbers quoted may improve over time. In the sequel `nproc` denotes the number of mpi processes (i.e. subdomains), while `nprol` denotes the number of shared-memory (OpenMP) cores used per mpi process/subdomain. The total number of cores employed is then given by `ncore=nproc*nprol`. The tables also quote the absolute (els/sec) and relative (els/sec/core) grid generation speeds achieved. Note that for perfect scaling, the relative grid generation speed should stay constant.

10.1 Garage

This example was taken from a blast simulation carried out for an office complex. The outline of the domain, as well as the trace of the domain defining grid partition on the surface is shown in Figure 3a. Figures 3a-c show the trace of the domain defining grid partition on the surface as well as the fronts after the parallel grid generation passes using 64 domains (mpi processors) for a finer mesh. These steps are shown in more detail in Figures 3d-k. Table 1 gives a compilation of timings for different mesh sizes, domains and processors on different machines. One may note that: a) Generating the 121 M mesh on one 8-core shared memory node (i.e. `nproc=1, nprol=8`) is slower than the distributed memory equivalent (i.e. `nproc=8, nprol=1`); b) The number of elements per core should exceed a minimum value (typically of the order of 2-4 Mels) in order to reach a generation speed per core that is acceptable; c) The local OMP scaling improves as the number of elements in each domain is increased; d) It only takes on the order of five minutes to generate a mesh of 121 Mels on 256 cores (`nproc=32, nprol=8`).



Figures 3a-c Garage: DDG and Parallel Grid Generation



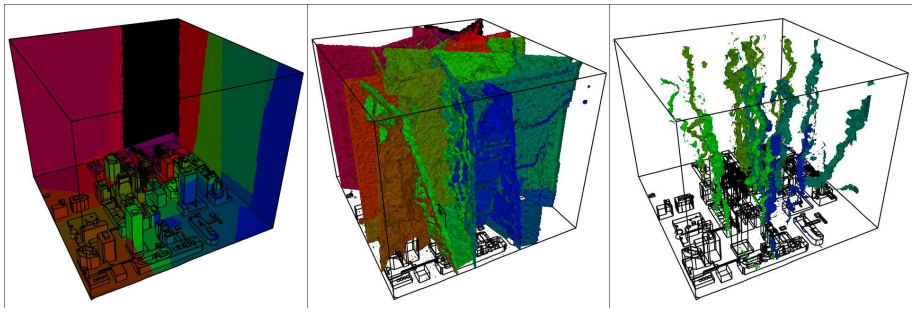
Figures 3d-k Garage: Front After Each Parallel Grid Generation Step

Machine	nproc	nprol	ncore	nelem	CPU [sec]	AbsSpeed [els/sec]	RelSpeed [els/sec/core]
Xeon(1)	1	8	8	120 M	2,293	52,333	6,542
SGI ITL	8	1	8	121 M	1,605	75,389	9,423
SGI ITL	8	8	64	121 M	516	234,496	3,664
Cry AMD	8	1	8	121 M	2,512	48,169	6,021
Cry AMD	16	1	16	121 M	1,954	61,924	3,870
Cry AMD	32	1	32	121 M	1,209	100,082	3,128
SGI ITL	32	8	256	121 M	316	383,293	1,497
Cry AMD	64	1	64	972 M	6,048	160,714	2,511
SGI ITL	64	8	512	1010 M	2,504	403,354	788

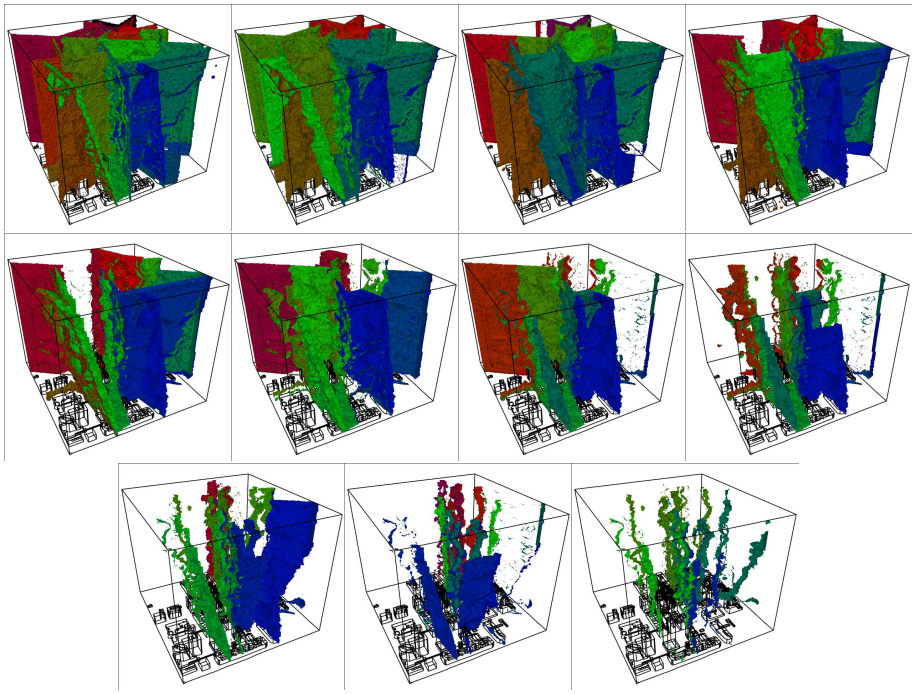
Table 1. Garage

10.2 Generic City Center

This example was taken from a recent blast and dispersion simulation. The outline of the domain, as well as the active front after each of the parallel grid generation passes for 32 processors (mpi domains) are shown in Figures 4a-c. Figures 4d-n show the active front after the generation passes in more detail. Table 2 gives a compilation of timings for different mesh sizes, domains and processors on different machines. One may observe the same general trends as observed for the previous case.



Figures 4a-c Generic City Center: Domain Defining Grid Partition and Parallel Grid Generation



Figures 3d-n Generic City Center: Parallel Grid Generation

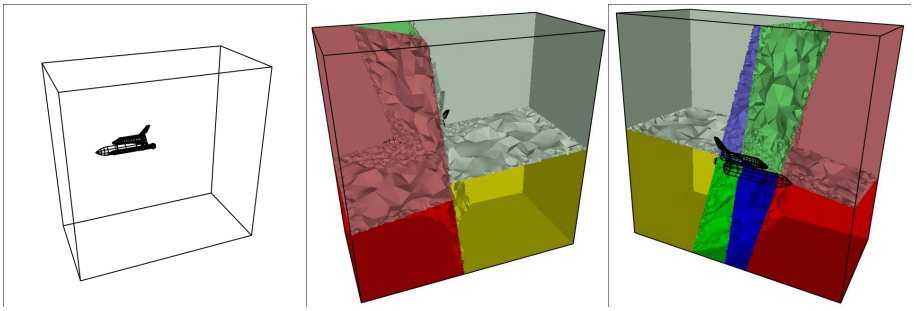
10.3 Shuttle Ascent Configuration

This example has been used repeatedly for benchmarking purposes. The outline of the domain may be seen in Figure 5a. The trace of the domain defining grid partition on the surface is shown in Figures 5b,c. Figures 5d,e show the active front after the first generation pass using 8 domains (mpi processors).

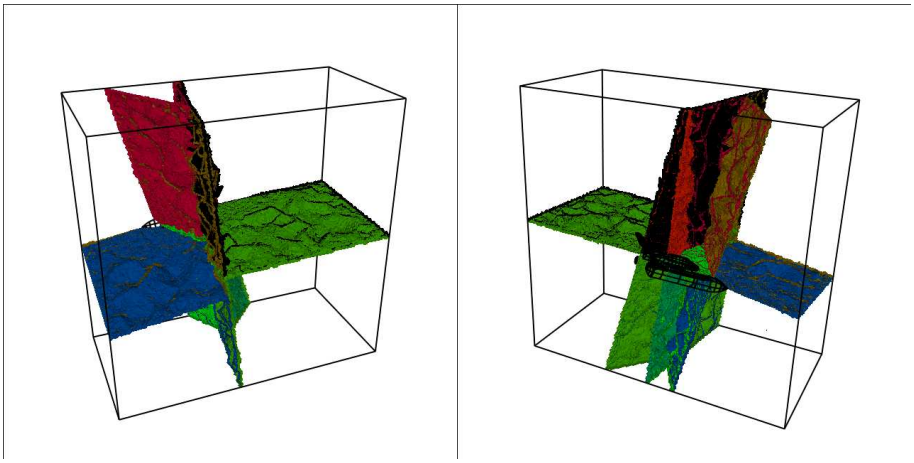
Machine	nproc	nprol	ncore	nelem	CPU [sec]	AbsSpeed [els/sec]	RelSpeed [els/sec/core]
Cry AMD	32	1	32	135 M	1,824	74,013	2,312
SGI ITL	16	8	128	135 M	556	242,805	1,897
SGI ITL	32	1	32	135 M	977	138,178	4,318
SGI ITL	32	2	64	135 M	754	179,045	2,797
SGI ITL	32	4	128	135 M	571	236,427	1,847
SGI ITL	32	8	256	135 M	488	276,639	1,080

Table 2. Generic City Center

This mesh had approximately 120 Mels. Table 3 gives a brief compilation of timings.



Figures 5a-c Shuttle: Outline of Domain and DDG Partition



Figures 5d,e Shuttle: Front After 1st Parallel Grid Generation Pass

Machine	nproc	nprol	ncore	nelem	CPU [sec]	AbsSpeed [els/sec]	RelSpeed [els/sec/core]
Xeon	8	1	8	27 M	872	30,963	3,870
Xeon	8	1	8	108 M	3,128	34,526	4,315

Table 3. Shuttle

11 General Observations

While the first parallel grid generation pass (i.e. generating elements inside each domain) scales perfectly, the scaling can degrade quikly for the subsequent passes (i.e. those that mesh the inter-domain boundary regions). The recourse taken here is to simply generate the remaning elements in one processor once the global number of remaning faces drops below 0.5 Mfaces. We are presently working on ways to mitigate this 'logarithmic trap'.

12 Conclusions and Outlook

A scalable, parallel advancing grid generation technique has been developed for complex geometries and meshes with large size variations. The key innovation compared to previous techniques is the use of a domain-defining grid that has the same fine surface triangulation as the final mesh desired, but a much coarser interior mesh. In this way, the domain to be gridded is uniquely defined, overcoming a shortcoming of previous approaches. This domain-defining grid is then partitioned according to the estimated number of elements to be generated, allowing for a balanced distribution of work among the processors. The domain defining grid is also used to redistribute the elements and points after grid generation, and during the subsequent mesh improvement.

Timings show that the proposed approach is scalable and able to produce large grids of high quality in a modest amount of clocktime.

With the proposed parallel grid generator, a major impediment to a completely scalable simulation pipeline (grid generation, solvers, post-processing) has been removed, opening the way for truly large-scale computations using unstructured, body-fitted grids.

Acknowledgements

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725, and also resources of the DoD High Performance Computing Modernization Program. This support is greatfully acknowledged.

References

1. A. Alleaume, L. Francez, M. Loriot and N. Maman - Large OutofCore Tetrahedral Meshing; *Proc. 16th International Meshing Roundtable*, Sandia National Laboratory, Oct. 15-17 (2007).
2. H. Andrae, E. Ivanov, O. Gluchshenko, A. Kudryavtsev - Automatic Parallel Generation of Tetrahedral Grids by Using a Domain Decomposition Approach; *J. Comp. Math. and Math. Phys.* 48, 8, 1448-1457 (2008).
3. T.J. Baker - Developments and Trends in Three-Dimensional Mesh Generation. *Appl. Num. Math.* **5**, 275-304 (1989).
4. J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of a Blast Inside a Boeing 747; *AIAA*-93-3091 (1993).
5. J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of Blast in the World Trade Center; *AIAA*-95-0085 (1995).
6. J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman - A Coupled Fluid/Structure Modeling of Shock Interaction with a Truck; *AIAA*-96-0795 (1996).
7. J.D. Baum, H. Luo and R. Löhner - The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies; *AIAA*-98-0788 (1998).
8. J.D. Baum, H. Luo, E. Mestreau, R. Löhner, D. Pelessone and C. Charman - A Coupled CFD/CSD Methodology for Modeling Weapon Detonation and Fragmentation; *AIAA*-99-0794 (1999).
9. G.E. Blelloch, J.C. Hardwick, G.L. Miller and D. Talmor - Design and Implementation of a Practical Parallel Delaunay Algorithm; *Algorithmica*, 24, 243-269 (1999).
10. L.P. Chew, N. Chrisochoides and F. Sukup - Parallel Constrained Delaunay Meshing; *Proc. 1997 Workshop on Trends in Unstructured Mesh Generation*, June (1997)
11. N. Chrisochoides and D. Nave - Simultaneous Mesh Generation and Partitioning for Delaunay Meshes; pp. 55-66 in *Proc. 8th Int. Meshing Roundtable*, South Lake Tahoe, October (1999).
12. N. Chrisochoides and D. Nave - Parallel Delaunay Mesh Generation Kernel; *Int. J. Num. Meth. Eng.* 58, 161-176 (2003).
13. N. Chrisochoides - Parallel Mesh Generation; pp 237-259 in *Numerical Solution of Partial Differential Equations on Parallel Computers*, (A.M. Bruaset, A. Tveito eds.), Springer (2005).
14. H.L. de Cougny, M.S. Shephard and C. Ozturan - Parallel Three-Dimensional Mesh Generation; *Computing Systems in Engineering* 5, 311-323 (1994).
15. H.L. de Cougny, M.S. Shephard and C. Ozturan - Parallel Three-Dimensional Mesh Generation on Distributed Memory MIMD Computers; *Tech. Rep. SCOREC Rep. # 7*, Rensselaer Polytechnic Institute (1995).
16. H. de Cougny and M.Shephard - Parallel Volume Meshing Using Face Removals and Hierarchical Repartitioning; *Comp. Meth. Appl. Mech. Eng.* 174(3-4), 275-298 (1999).
17. E. Darve and R. Löhner - Advanced Structured-Unstructured Solver for Electromagnetic Scattering from Multimaterial Objects; *AIAA*-97-0863 (1997).
18. L.A. Freitag and C.-Ollivier Gooch - Tetrahedral Mesh Improvement Using Swapping and Smoothing; *Int. J. Num. Meth. Eng.* , 40, 3979-4002 (1997).

19. J. Frykestig - Advancing Front Mesh Generation Techniques with Application to the Finite Element Method; *Pub. 94:10*, Chalmers University of Technology; Göteborg, Sweden (1994).
20. J. Galtier and P.L. George - Prepartitioning as a Way to Mesh Subdomains in Parallel; *Special Symposium on Trends in Unstructured Mesh Generation* 107-122, ASME/ASCE/SES (1997).
21. P.L. George, F. Hecht and E. Saltel - Automatic Mesh Generator With Specified Boundary; *Comp. Meth. Appl. Mech. Eng.* 92, 269-288 (1991).
22. P.L. George - Tet Meshing: Construction, Optimization and Adaptation; *Proc. 8th Int. Meshing Roundtable*, South Lake Tahoe, October (1999).
23. R. von Hanxleden and L.R. Scott - Load Balancing on Message Passing Architectures; *J. Parallel and Distr. Comp.* 13, 312-324 (1991).
24. O. Hassan, L.B. Bayne, K. Morgan and N. P. Weatherill - An Adaptive Unstructured Mesh Method for Transient Flows Involving Moving Boundaries; pp. 662-674 in *Computational Fluid Dynamics '98* (K.D. Papailiou, D. Tsahalis, J. Périaux and D. Knörzer eds.) Wiley (1998).
25. Y. Ito, A.M. Shih, A.K. Erukala, B.K. Soni, A. Chernikov, N. Chrisochoides and K. Nakahashi - Parallel Unstructured Mesh Generation by an Advancing Front Method; *J. Mathematics and Computers in Simulation* 75, 5-6, 200-209 (2007).
26. E.G. Ivanov, H. Andrae and A.N. Kudryavtsev - Domain Decomposition Approach for Automatic Parallel Generation of Tetrahedral Grids;*Int. Math. J. Comp. Meth. in App. Math.* 6, 2, 178-193 (2006).
27. H. Jin and R.I. Tanner - Generation of Unstructured Tetrahedral Meshes by the Advancing Front Technique; *Int. J. Num. Meth. Eng.* 36, 1805-1823 (1993).
28. W. Jou - Comments on the Feasibility of LES for Commercial Airplane Wings; *AIAA*-98-2801 (1998).
29. C. Kadow and N. Walkington - Design of a Projection-Based Parallel Delaunay Mesh Generation and Refinement Algorithm; *Proc. Fourth Symp. on Trends in Unstructured Mesh Generation* (2003).
30. A. Kamoulakos, V. Chen, E. Mestreau and R. Löhner - Finite Element Modelling of Fluid/ Structure Interaction in Explosively Loaded Aircraft Fuselage Panels Using PAMSHOCK/ PAMFLOW Coupling; *Conf. on Spacecraft Structures, Materials and Mechanical Testing*, Noordwijk, The Netherlands, March (1996).
31. G. Karypis and V. Kumar - A Parallel Algorithm for Multilevel Graph Partitioning and Sparse Matrix Ordering; *J. of Parallel and Distributed Computing* 48, 71 - 85 (1998).
32. G. Karypis and V. Kumar - Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs; *SIAM Review* 41, 2, 278 - 300 (1999).
33. B.G. Larwood, N.P. Weatherill, O. Hassan and K. Morgan - Domain Decomposition Approach for Parallel Unstructured Mesh Generation; *Int. J. Num. Meth. Eng.* 58, 2, 177-188 (2003).
34. J. Liu, K. Kailasanath, R. Ramamurti, D. Munday, E. Gutmark and R. Löhner - Large-Eddy Simulations of a Supersonic Jet and Its Near-Field Acoustic Properties; *AIAA J.* 47, 8, 1849-1864 (2009).
35. R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* 4, 123-135 (1988).
36. R. Löhner and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* 8, 1135-1149 (1988).

37. R. Löhner - Three-Dimensional Fluid-Structure Interaction Using a Finite Element Solver and Adaptive Remeshing; *Comp. Sys. in Eng.* 1, 2-4, 257-272 (1990).
38. R. Löhner, J. Camberos and M. Merriam - Parallel Unstructured Grid Generation; *Comp. Meth. Appl. Mech. Eng.* 95, 343-357 (1992).
39. R. Löhner and R. Ramamurti - A Load Balancing Algorithm for Unstructured Grids; *Comp. Fluid Dyn.* 5, 39-58 (1995).
40. R. Löhner - Extensions and Improvements of the Advancing Front Grid Generation Technique; *Comm. Num. Meth. Eng.* 12, 683-702 (1996).
41. R. Löhner - Regridding Surface Triangulations; *J. Comp. Phys.* 126, 1-10 (1996).
42. R. Löhner - Progress in Grid Generation via the Advancing Front Technique; *Engineering with Computers* 12, 186-210 (1996).
43. R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure-Thermal Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA*-98-2419
44. (1998).
45. R. Löhner - Renumbering Strategies for Unstructured- Grid Solvers Operating on Shared- Memory, Cache- Based Parallel Machines; *Comp. Meth. Appl. Mech. Eng.* 163, 95-109 (1998).
46. R. Löhner, C. Yang and E. Oñate - Viscous Free Surface Hydrodynamics Using Unstructured Grids; *Proc. 22nd Symp. Naval Hydrodynamics*, Washington, D.C., August (1998).
47. R. Löhner - A Parallel Advancing Front Grid Generation Scheme; *Int. J. Num. Meth. Eng.* 51, 663-678 (2001).
48. R. Löhner - *Applied CFD Techniques, 2nd Edition*; J. Wiley & Sons (2008).
49. R. Löhner, J.R. Cebral, F.F. Camelli, S. Appanaboyina, J.D. Baum, E.L. Mestreau and O. Soto - Adaptive Embedded and Immersed Unstructured Grid Techniques; *Comp. Meth. Appl. Mech. Eng.* 197, 2173-2197 (2008).
50. D.L. Marcum and N.P. Weatherill - Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection; *AIAA J.* 33, 9, 1619-1625 (1995).
51. D.J. Mavriplis and S. Pirzadeh - Large-Scale Parallel Unstructured Mesh Computations for 3-D High-Lift Analysis; *ICASE Rep.* 99-9 (1999).
52. E. Mestreau, R. Löhner and S. Aita - TGV Tunnel-Entry Simulations Using a Finite Element Code with Automatic Remeshing; *AIAA*-93-0890 (1993).
53. E. Mestreau and R. Löhner - Airbag Simulation Using Fluid/Structure Coupling; *AIAA*-96-0798 (1996).
54. K. Morgan, P.J. Brookes, O. Hassan and N.P. Weatherill - Parallel Processing for the Simulation of Problems Involving Scattering of Electro-Magnetic Waves; in *Proc. Symp. Advances in Computational Mechanics* (L. Demkowicz and J.N. Reddy eds.) (1997).
55. T. Okusanya and J. Peraire - Parallel Unstructured Mesh Generation; *Proc. 5th Int. Conf. Num. Grid Generation in CFD and Related Fields*, Mississippi, April (1996).
56. T. Okusanya and J. Peraire - 3-D Parallel Unstructured Mesh Generation; *Proc. Joint ASME/ASCE/SES Summer Meeting* (1997).
57. J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz - Adaptive Remeshing for Compressible Flow Computations; *J. Comp. Phys.* 72, 449-466 (1987).
58. J. Peraire, J. Peiro, L. Formaggia K. Morgan and O.C. Zienkiewicz - Finite Element Euler Calculations in Three Dimensions; *Int. J. Num. Meth. Eng.* 26, 2135-2159 (1988).

59. J. Peraire, K. Morgan and J. Peiro - Unstructured Finite Element Mesh Generation and Adaptive Procedures for CFD; *AGARD-CP-464*, 18 (1990).
60. J. Peraire, K. Morgan, and J. Peiro - Adaptive Remeshing in 3-D; *J. Comp. Phys.* (1992).
61. R. Said, N.P. Weatherill, K. Morgan and N.A. Verhoeven - Distributed Parallel Delaunay Mesh Generation; to appear in *Comp. Meth. Appl. Mech. Eng.* (1999).
62. A. Shostko and R. Löhner - Three-Dimensional Parallel Unstructured Grid Generation; *Int. J. Num. Meth. Eng.* 38, 905-925 (1995).
63. R. Tilch, A. Tabbal, M. Zhu, F. Decker and R. Löhner - Combination of Body-Fitted and Embedded Grids for External Vehicle Aerodynamics; *Engineering Computations* 25, 1, 28-41 (2008).
64. U. Tremel, K.A. Sorensen, S. Hitzel, H. Rieger, O. Hassan and N.P. Weatherill - Parallel Remeshing of Unstructured Volume Grids for CFD Applications; *Int. J. Num. Meth. Fluids* 53, 8, 1361-1379 (2006).
65. A. Vidwans, Y. Kallinderis and V. Venkatakrishnan - A Parallel Load Balancing Algorithm for 3-D Adaptive Unstructured Grids; *AIAA*-93-3313-CP (1993).
66. D. Williams - Performance of Dynamic Load Balancing Algorithms for Unstructured Grid Calculations; *CalTech Rep.* C3P913 (1990).
67. N.P. Weatherill - Delaunay Triangulation in Computational Fluid Dynamics; *Comp. Math. Appl.* 24, 5/6, 129-150 (1992).
68. N.P. Weatherill and O. Hassan - Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints; *Int. J. Num. Meth. Eng.* 37, 2005-2039 (1994).
69. S. Yoshimura, H. Nitta, G. Yagawa and H. Akiba - Parallel Automatic Mesh Generation Method of Ten-Million Nodes Problem Using Fuzzy Knowledge Processing and Computational Geometry; *Proc. 4th World CongĊomp. Mech.* Buenos Aires, Argentina, July (1998).